

PRACTICAL NO. 3 Remote Object Communication

a) Using MySQL create Library database. Create table Book (Book_id, Book_name, Book_author) and retrieve the Book information from Library database using Remote Object Communication concept.

Code:

LibraryInterface.java

```
package libDB;
import java.rmi.Remote;
import java.rmi.RemoteException;
import java.util.List;
public interface LibraryInterface extends Remote {
    List<String> getAllBooks() throws RemoteException;
    String getBookById(int id) throws RemoteException;
}
```

LibraryImpl.java

```
package libDB;
import java.rmi.RemoteException;
import java.rmi.server.UnicastRemoteObject;
import java.sql.Connection;
import java.sql.Driver;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.util.ArrayList;
import java.util.List;
import java.sql.Statement;
public class LibraryImpl extends UnicastRemoteObject implements LibraryInterface {

    Connection conn;

    protected LibraryImpl() throws RemoteException
    {
        super();
        try {
            //connect to mysql
            conn = DriverManager.getConnection("jdbc:mysql://localhost:3306/prac3","root","");

        } catch (SQLException e) {
            e.printStackTrace();
        }
    }

    @Override
    public List<String> getAllBooks() throws RemoteException{
        List<String> books = new ArrayList<String>();
```

```

        try {
            Statement stmt= conn.createStatement();
            ResultSet rs = stmt.executeQuery("SELECT * FROM Book");
            while(rs.next())
            {
                books.add(rs.getInt(1)+" - "+rs.getString(2)+" by "+rs.getString(3));
            }
        } catch (SQLException e) {
            e.printStackTrace();
        }
        return books;
    }

    @Override
    public String getBookById(int id) throws RemoteException{
        try {
            PreparedStatement ps= conn.prepareStatement("SELECT * FROM Book WHERE
Book_id = ?");
            ps.setInt(1, id);
            ResultSet rs = ps.executeQuery();
            if (rs.next()) {
                return rs.getInt(1) + " - " + rs.getString(2)+" by" + rs.getString(3);
            }
        } catch (SQLException e) {
            // TODO: handle exception
            e.printStackTrace();
        }
        return "Book not found";
    }
}

```

LibraryServer.java

```

package libDB;
import java.rmi.registry.LocateRegistry;
import java.rmi.registry.Registry;
public class LibraryServer {
    public static void main(String[] args) {
        try {
            LibraryImpl obj = new LibraryImpl();
            Registry reg = LocateRegistry.createRegistry(1090);
            reg.rebind("LibraryService", obj);
            System.out.println("Library RMI Server is running....");
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}

```

LibraryClient.java

```

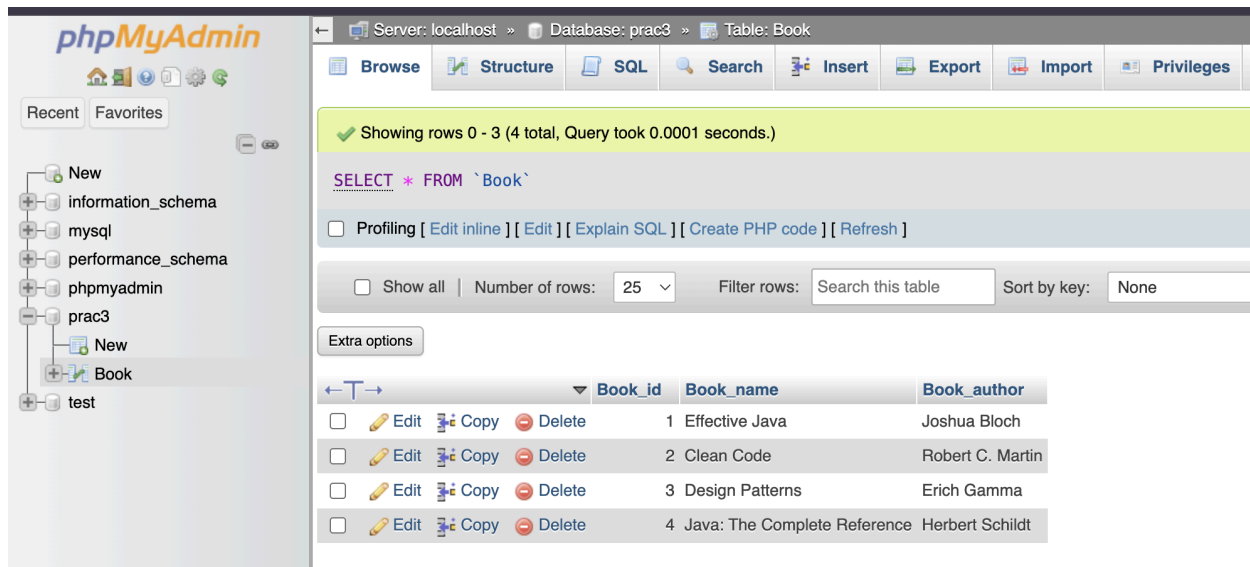
package libDB;
import java.rmi.registry.LocateRegistry;
import java.rmi.registry.Registry;
import java.util.Iterator;
import java.util.List;
import java.util.Scanner;
public class LibraryClient {
    public static void main(String[] args) {
        try {
            Registry reg = LocateRegistry.getRegistry("localhost",1090);
            LibraryInterface lib= (LibraryInterface) reg.lookup("LibraryService");

            Scanner sc= new Scanner(System.in);
            System.out.println("1. get All books ");
            System.out.println("2. get Books by ID ");
            System.out.print("Enter choice: ");
            int choice = sc.nextInt();

            if (choice == 1) {
                List<String> books = lib.getAllBooks();
                for(String b : books)
                {
                    System.out.println(b);
                }
            }
            else if (choice ==2) {
                System.out.println("Enter Book ID : ");
                int id = sc.nextInt();
                System.out.println(lib.getBookById(id));
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}

```

Output



Server: localhost » Database: prac3 » Table: Book

Showing rows 0 - 3 (4 total, Query took 0.0001 seconds.)

`SELECT * FROM `Book``

☐ Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP code] [Refresh]

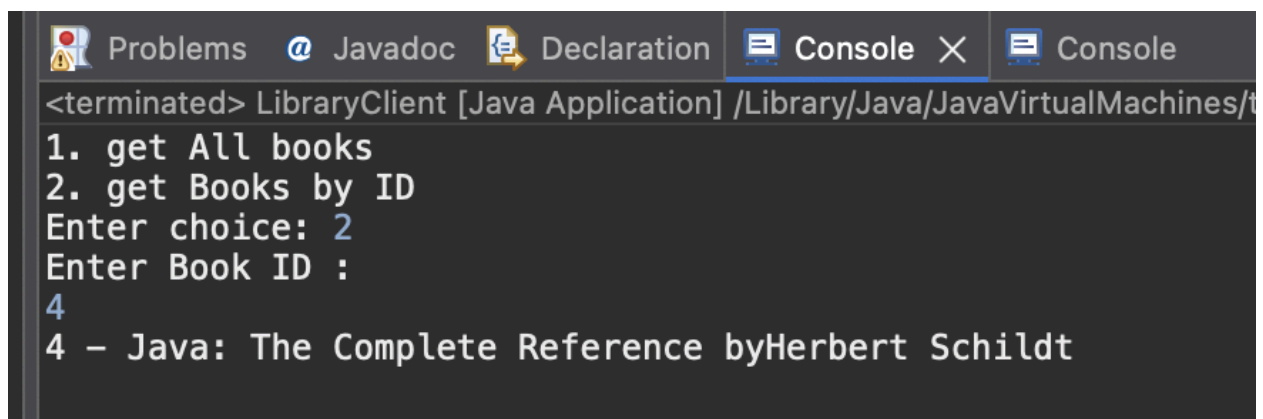
☐ Show all | Number of rows: 25 | Filter rows: Search this table | Sort by key: None

Extra options

	Book_id	Book_name	Book_author
<input type="checkbox"/> Edit <input type="checkbox"/> Copy <input type="checkbox"/> Delete	1	Effective Java	Joshua Bloch
<input type="checkbox"/> Edit <input type="checkbox"/> Copy <input type="checkbox"/> Delete	2	Clean Code	Robert C. Martin
<input type="checkbox"/> Edit <input type="checkbox"/> Copy <input type="checkbox"/> Delete	3	Design Patterns	Erich Gamma
<input type="checkbox"/> Edit <input type="checkbox"/> Copy <input type="checkbox"/> Delete	4	Java: The Complete Reference	Herbert Schildt

Library RMI Server is running....

```
<terminated> LibraryClient [Java Application] /Library/Java/JavaVirtualMachines/temu
1. get All books
2. get Books by ID
Enter choice: 1
1 - Effective Java by Joshua Bloch
2 - Clean Code by Robert C. Martin
3 - Design Patterns by Erich Gamma
4 - Java: The Complete Reference by Herbert Schildt
```



Problems Javadoc Declaration Console Console

```
<terminated> LibraryClient [Java Application] /Library/Java/JavaVirtualMachines/t
1. get All books
2. get Books by ID
Enter choice: 2
Enter Book ID :
4
4 - Java: The Complete Reference byHerbert Schildt
```

b) Using MySQL create Electric_Bill database. Create table Bill (consumer_name, bill_due_date, bill_amount) and retrieve the bill information from the Electric_Bill database using Remote Object Communication concept.

Code:**BillService.java**

```
package BillDB;
import java.rmi.Remote;
import java.rmi.RemoteException;
import java.util.List;
public interface BillService extends Remote {
    List<String> getBillInfo() throws RemoteException;
}
```

BillServiceImpl.java

```
package BillDB;
import java.rmi.server.UnicastRemoteObject;
import java.rmi.RemoteException;
import java.sql.*;
import java.util.ArrayList;
import java.util.List;
public class BillServiceImpl extends UnicastRemoteObject implements BillService {
    protected BillServiceImpl() throws RemoteException {
        super();
    }
    @Override
    public List<String> getBillInfo() throws RemoteException {
        List<String> bills = new ArrayList<>();
        try {
            Class.forName("com.mysql.cj.jdbc.Driver");
            Connection con = DriverManager.getConnection(
                "jdbc:mysql://localhost:3306/Electric_Bill", "root", "password");
            Statement stmt = con.createStatement();
            ResultSet rs = stmt.executeQuery("SELECT * FROM Bill");
            while (rs.next()) {
                String row = "Consumer: " + rs.getString("consumer_name") +
                    ", Due Date: " + rs.getDate("bill_due_date") +
                    ", Amount: " + rs.getBigDecimal("bill_amount");
                bills.add(row);
            }
            con.close();
        } catch (Exception e) {
            e.printStackTrace();
        }
        return bills;
    }
}
```

BillServer.java

```
package BillDB;
```

```

import java.rmi.registry.LocateRegistry;
import java.rmi.registry.Registry;
public class BillServer {
    public static void main(String[] args) {
        try {
            BillServiceImpl obj = new BillServiceImpl();
            Registry reg = LocateRegistry.createRegistry(1099);
            reg.rebind("BillService", obj);
            System.out.println("Bill RMI Server is running...");
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}

```

BillClient.java

```

package BillDB;
import java.rmi.registry.LocateRegistry;
import java.rmi.registry.Registry;
import java.util.List;
import java.util.Scanner;
public class BillClient {
    public static void main(String[] args) {
        try {
            // Connect to RMI registry
            Registry reg = LocateRegistry.getRegistry("localhost", 1099);
            BillService billService = (BillService) reg.lookup("BillService");
            Scanner sc = new Scanner(System.in);
            int choice;
            do {
                System.out.println("\n===== Electric Bill Menu =====");
                System.out.println("1. Get All Bills");
                System.out.println("2. Get Bill by Consumer Name");
                System.out.print("Enter your choice: ");
                choice = sc.nextInt();
                sc.nextLine(); // consume newline
                switch (choice) {
                    case 1:
                        List<String> bills = billService.getAllBills();
                        if (bills.isEmpty()) {
                            System.out.println("No bills found!");
                        } else {
                            System.out.println("\n--- Bill Details ---");
                            for (String b : bills) {
                                System.out.println(b);
                            }
                        }
                        break;
                    case 2:

```

```

        System.out.print("Enter Consumer Name: ");
        String name = sc.nextLine();
        String result = billService.getBillByConsumer(name);
        System.out.println("\n" + result);
        break;
    default:
        System.out.println("Invalid choice. Try again!");
    }
} while (choice != 3);
sc.close();
} catch (Exception e) {
    e.printStackTrace();
}
}
}

```

Output:

The screenshot shows the phpMyAdmin interface. On the left is the database structure tree. The main panel displays the 'Bill' table with the following data:

consumer_name	bill_due_date	bill_amount
John Doe	2025-09-15	1200.50
Alice Smith	2025-09-20	980.75
Bob Lee	2025-09-25	1430.00

The screenshot shows an IDE console window with the following output:

```

BillServer [Java Application] /Library/Java/JavaVirtualMachines/temurin-21.jdk/Cont
Bill RMI Server is running...

```

```
===== Electric Bill Menu =====
1. Get All Bills
2. Get Bill by Consumer Name
Enter your choice: 1
|
--- Bill Details ---
John Doe | Due: 2025-09-15 | Amount: 1200.50
Alice Smith | Due: 2025-09-20 | Amount: 980.75
Bob Lee | Due: 2025-09-25 | Amount: 1430.00
```

```
===== Electric Bill Menu =====
1. Get All Bills
2. Get Bill by Consumer Name
Enter your choice: 2
Enter Consumer Name: john doe
|
John Doe | Due: 2025-09-15 | Amount: 1200.50
```

Conclusion:

This practical helped us understand and implement Java Remote Method Invocation (RMI) to communicate between client and server applications. We successfully created databases (Library and Electric_Bill), connected them using **JDBC**, and retrieved records remotely. This demonstrates how distributed applications can share data securely and efficiently across different machines.